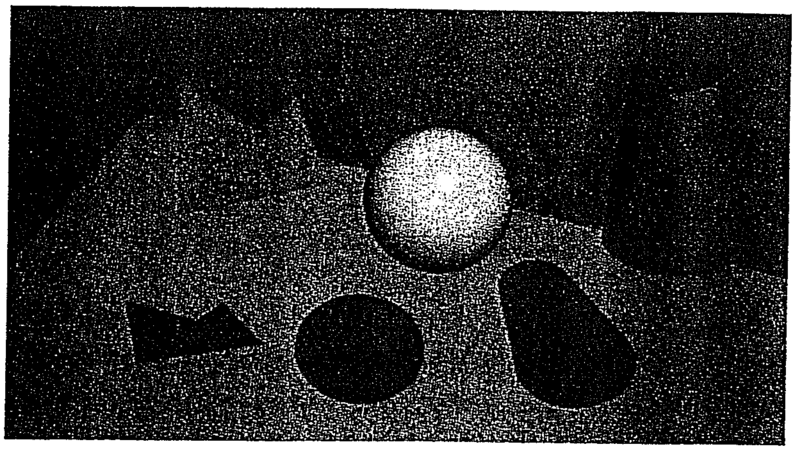
LBL-37865



VISUALIZATION

Gregory J. Ward

Two approaches to the design of lighting systems have developed over the years, embodied in the respective roles of the lighting designer and the illuminating engineer. The illuminating engineer relies on lookup tables and calculations to make lighting decisions, emphasizing the importance of light level over most other considerations. The lighting designer, in contrast, is guided by experience and aesthetic sense to illuminate environments in a way pleasing to the eye. For the illuminating engineer, the current revolution in computer technology has brought accurate illuminance predictions for simple spaces, making selection more reliable and, some would argue, easier. However, the computer revolution has done little to assist the lighting designer, whose evaluation more often depends on visual qualities than on numerical quantities.

The emerging field of visualization in computer science combines calculations with computer graphics to bring another dimension of understanding to scientists, engineers, and designers. This new development holds particular promise for lighting systems design, which is both a numerical and a visual endeavor. Rather than being restricted as an engineer to a lighting decision based on a table of illuminance values, or as a designer relying only on experience, one could refer to a computer simulation that predicts quantity and displays quality.

For such a visualization tool to be valuable to the lighting designer, it must model all the complexity found in real designs, including light fixtures, surface geometries, materials, and daylight. Unfortunately, currently available lighting software is limited mostly to empty rectangular rooms with matte surfaces. The programs use flux exchange methods which rely on diffuse reflection and simple geometry to operate efficiently,[1] and produce synthetic images that contain little more information than the illuminance tables from which they are projected.[2] A different simulation technique, called backwards ray tracing, is suited well to computer graphics and visualization.[3] The path of light is tracked from its presumed destination to one or more sources, taking into account specular reflection, transmission, and virtually any geometry. In this way, ray tracing calculates luminance directly, which is ideal for the visualization of illuminated spaces as an image is really just a collection of luminance values. Recent advances in this technique have resulted in a complete lighting calculation that incorporates diffuse interreflection and daylight.[4,5]

An efficient ray tracing package, called RADIANCE, has been developed at Lawrence Berkeley Laboratories for accurate luminance prediction, and is now ready for distribution. The program takes a scene description with light sources, sun, sky, buildings, rooms, furniture, etc. and produces spectral radiance values which can be collected in a color image. As a lighting design tool, RADIANCE represents a significant advance in the state of the art, and it is being made available to interested parties for free.

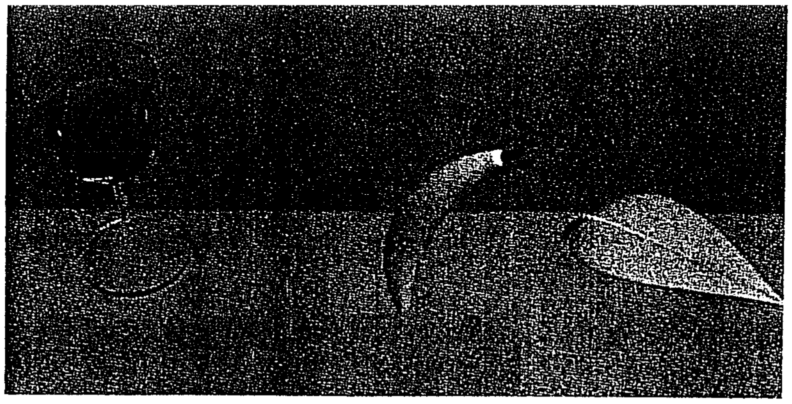
RADIANCE

RADIANCE was written initially to explore new and old techniques in lighting simulation. As a research tool, it lacks many of the user-friendly features found in commercial software packages. However, RADIANCE is more versatile than other lighting simulations and faster than other ray tracing programs, with some important capabilities not found in either. Foremost is the ability to accurately account for both diffuse and specular interreflection in complicated spaces. This is a basic requirement for the prediction of luminance in architectural applications.

RADIANCE typically operates in conjunction with a commercial CAD system, which is used to create the scene geometry. An import program takes the CAD scene description and converts it to the RADIANCE input format. Details such as surface properties are added, analysis is performed, and design modifications are made based on the result.

General

RADIANCE is currently available for minicomputers, workstations, and high-end personal computers that run the UNIX operating system. The software is divided into several components, whose relationships are shown in **Figure 1**. In the center are the three main programs, RTRACE, RPICT, and



RADIANCE: A software program for computing luminance and synthetic images

RVIEW. RPICT produces picture files in batch mode, and RVIEW displays scenes interactively. RTRACE calculates luminance, illuminance, and other application-specific information. The main programs are variations of the same basic ray tracer. Their input consists of scene files containing surface and material descriptions, OCTREE files produced by sorting this information (using OCONV), and auxiliary files containing information such as surface textures and candlepower distributions. Pictures produced by the programs can be displayed or sent to hardcopy devices such as film recorders and printers.

The interface to RADIANCE is command-based for maximum flexibility. Input files are created by CAD programs, text editors, and object generator programs. A mouse-based geometric editor is under development. The XFORM command permits arbitrary scaling and placement of objects in a hierarchical scene description. There is a library of useful generators to make prisms, patches, etc. and objects such as furniture and light sources are provided. The UNIX utility can be used to automate scene creation and rendering, and several other bundled programs simplify the creation of custom generators. Import programs for a few popular CAD systems will be provided.

Surfaces

At the lowest level, RADIANCE models polygons, spheres, and cones (Figure 2). From these basic boundary representations, varied and complicated shapes can be produced (Figure 3).

Continued on page 14

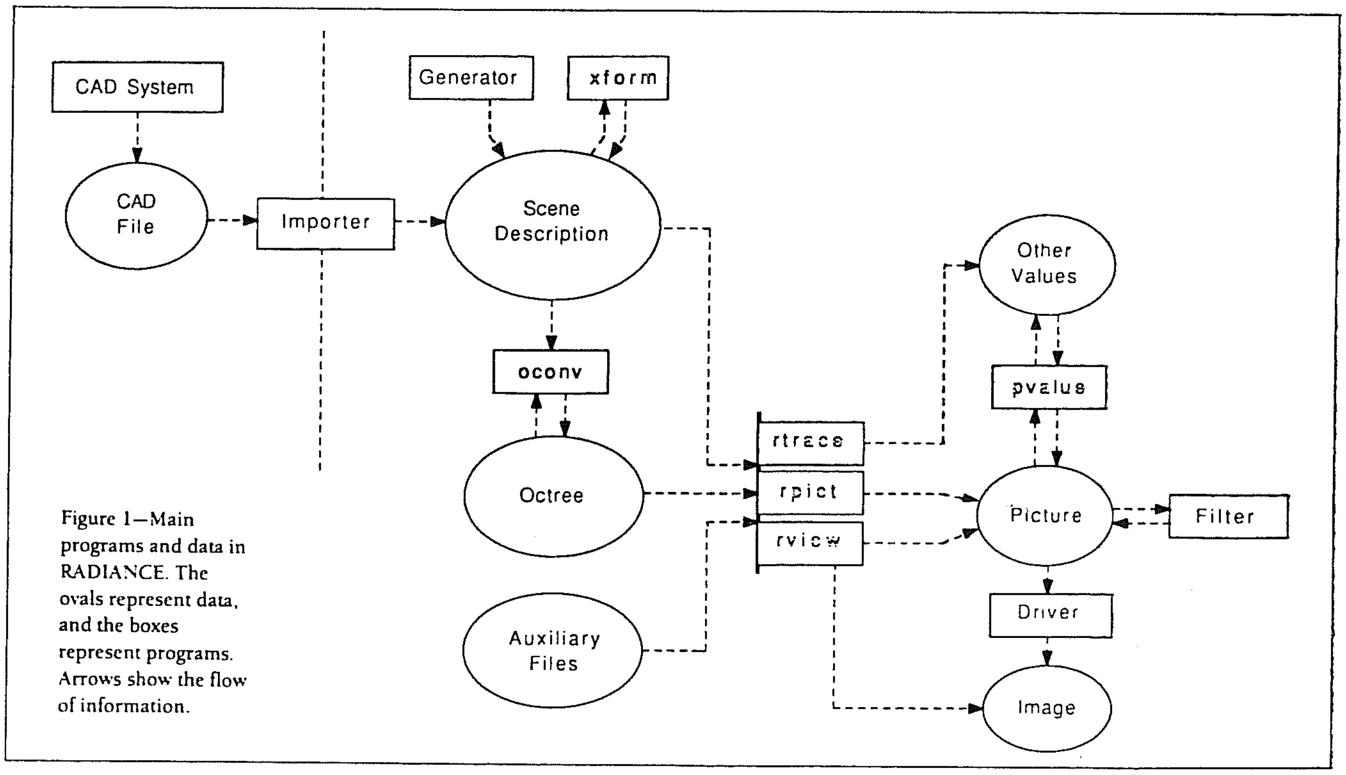


Figure 1—Main programs and data in RADIANCE. The ovals represent data, and the boxes represent programs. Arrows show the flow of information.

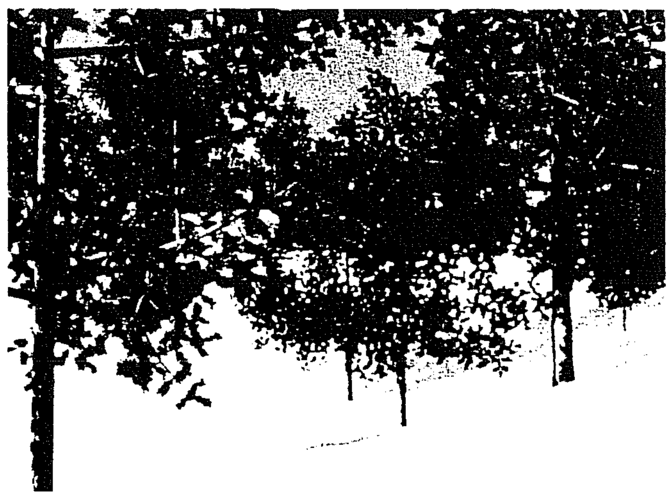


Figure 4—A forest scene modelled with more than 3 million surfaces. A single spray was instanced and combined with larger branches to form a tree, then the tree itself was instanced many times throughout the scene.

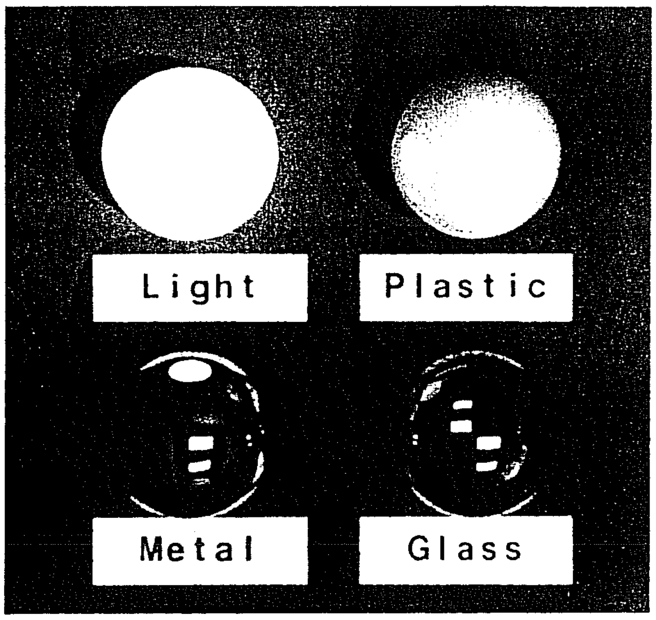


Figure 5—Some basic material types.

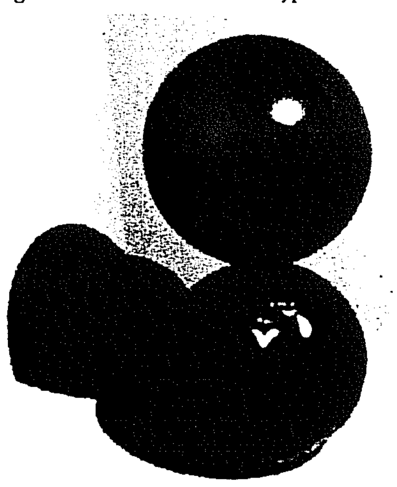


Figure 6—Pattern vs texture. The top ball has a procedural wood pattern, which is a variation in brightness only. The bottom ball has a similar texture, which instead affects the surface normal direction.

RADIANCE *from page 5*

Scenes are composed of many objects, and these objects may in turn contain many objects. This hierarchy is constructed through a simple mechanism: command expansion. When the program comes across a command in a scene file, it executes the command and interprets the output as more scene input. In a hierarchy, the XFORM command is used to read other scene files, transforming them to new positions. These scene files may contain other XFORM commands, thus producing a tree of transformations. XFORM can also be used to create arrays of objects, such as furniture or light sources. The ability to have a command within an scene file also simplifies the use of generator programs. Rather than including the output of a program that produces a box, for example, the command itself can be included in the scene file. Changes are straightforward, and the scene description is more compact. If desired, XFORM may be used to expand all commands, creating a flat-scene description.

Although the limit to the number of surfaces in the expanded geometric model is large—around 30,000, depending on memory—it is not infinite. To permit the modeling of more complex geometries, an instance structure is introduced. If a scene contains a few objects repeated many times throughout the space, instances can share the data of those objects, requiring memory for only individual transformations (Figure 4). Using this type of hierarchical instancing, scenes with hundreds of billions of primitives can be modeled.[6] Although rendering time does increase with the scene complexity, it is a sub-linear relationship $O(n^{1/3})$ such that an image with 1000 surfaces takes roughly twice as long as a scene with 125 surfaces, and a scene with 8000 surfaces takes four times as long.

Materials

Once the scene geometry has been described, the user must assign to each surface a material name, whose definition will determine how that surface interacts with light. RADIANCE provides several material types for this purpose (Figure 5). The basic type of emissive surfaces source is simply called light. Variations are provided for efficient modeling of spotlights, secondary emitters, and weak sources. Surfaces that reflect light are made of metal or plastic, and have both specular and diffuse components. Materials that transmit and refract light are also represented; glass is a common example. Within each material type, a broad range of light interactions can be simulated. For example, plastic can take on the appearance of any paint, paper, ceramic, or wood simply by varying parameters and applying optional textures and patterns.

Textures and patterns

Textures and patterns add important visual detail to surfaces without adding substantially to the model complexity. We define a texture as a perturbation of the surface normal, and a pattern as a perturbation of the material color. Figure 6 shows the effects of each on wood-colored plastic. A texture affects the illumination and highlights; whereas a pattern affects the reflectance.

RADIANCE provides several means for pattern specifica-

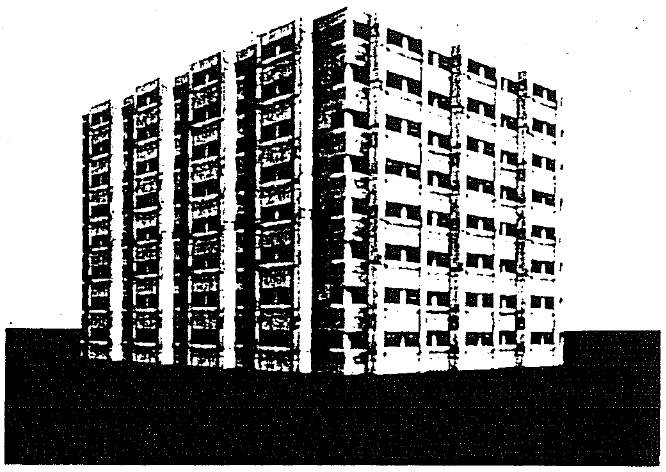


Figure 7—Scanned patterns. The building was made from images of a few windows duplicated on a simple structure and a brick pattern.

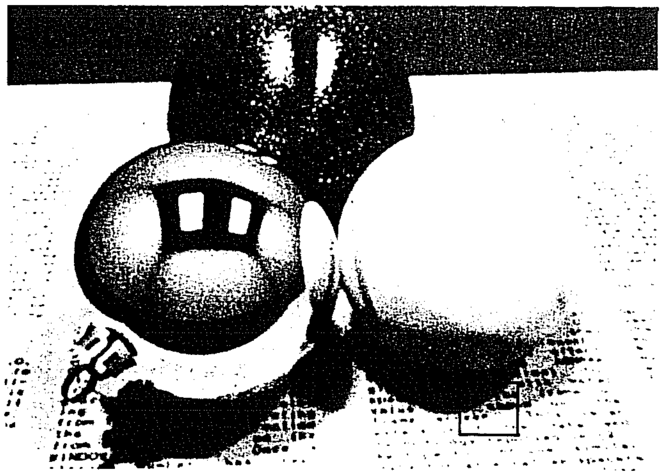


Figure 8—Orange peel texture and text pattern. The pixel values in the boxed area are plotted in Figure 11.

tions. A procedural pattern or texture is given as a formula that defines the pattern's value in terms of the current intersection point, ray direction, surface normal, distance, etc. The pattern in Figure 6 is related only to the three-dimensional intersection point, thus the ball appears as if it were cut out of solid wood. Patterns may also be scanned from photographs or video. Figure 7 has patterns that were digitized by a 35-mm slide scanner into picture files, then mapped onto the surfaces during rendering. The coordinate mapping for a pattern is determined by the user, and mappings from rectangular images to rectangles, cylinders, and spheres are provided. Figure 8 shows a procedural orange-peel texture and the built-in text pattern type. This illustration was inspired by an article on qualitative illumination.[7] Textures and patterns can be combined in almost any way imaginable.

Light sources

Because light sources are critical to the illumination of a scene, they are given special attention by the program. In general, a light source is differentiated from other surfaces by its material type. Currently polygons, spheres, and disks may be used for local sources; area sources and odd shapes may be modeled with polygonal meshes. Distant light sources such as the sun are modeled as a special case. A non-Lambertian source distribution is modeled with a pattern whose coordinates are mapped from the output direction. A conversion from the IESNA standard format for luminaire data will be provided.[8] If near-field photometry is available, the distribution may even be made to change over the area of the source.

Output

The most frequently desired output, and the type for which RADIANCE is tailored, is the color image. A color image is a two-dimensional array of radiance values rendered by projection from three-dimensions. The values are broken into spectral components, nominally red, green and blue. The renderer RPICT produces picture files in batch mode, and RVIEW calculates and displays images interactively.

Interactive image generation is an important feature of RADIANCE that permits quick error checking, view determination, and lighting evaluation. Few things are more frustrating than waiting several hours for an image, only to discover errors in the input (Figure 9). In a matter of minutes, RVIEW produces a low-resolution image clear enough to spot any

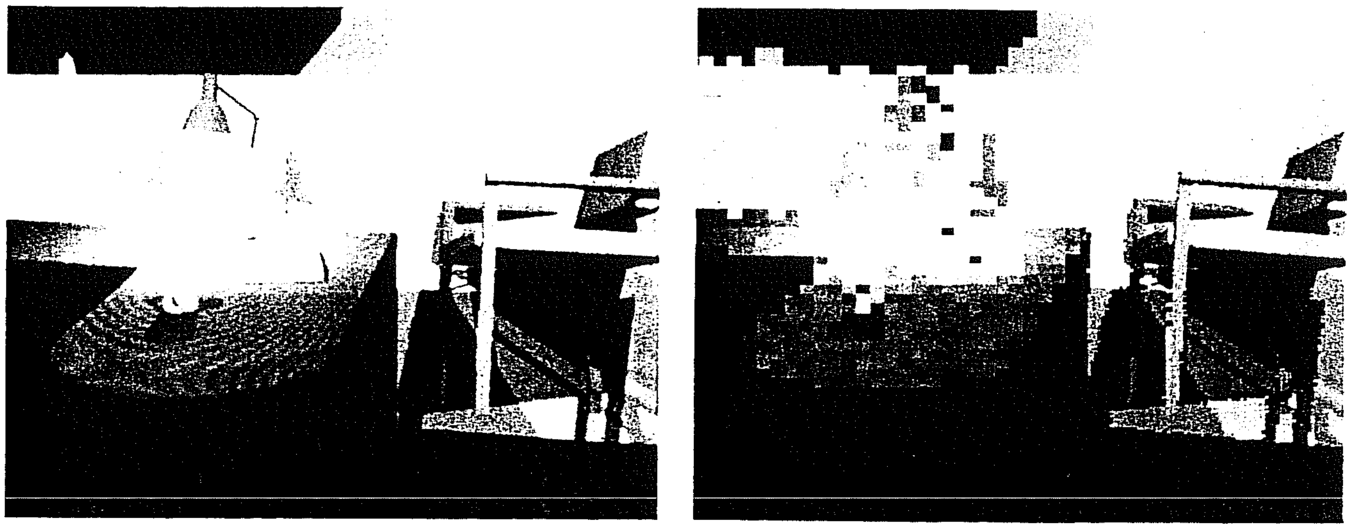


Figure 9—Finding errors in a scene description. An expensive rendering of a chair in a wall produced by RPICT, and an inexpensive rendering by RVIEW.

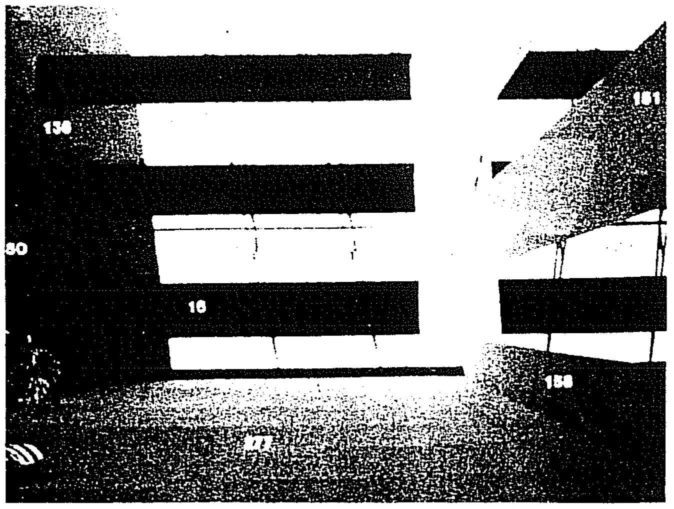


Figure 10—An atrium with direct sun. The superimposed luminance values (in nits) sample a tiny fraction of the numerical information contained in the picture.

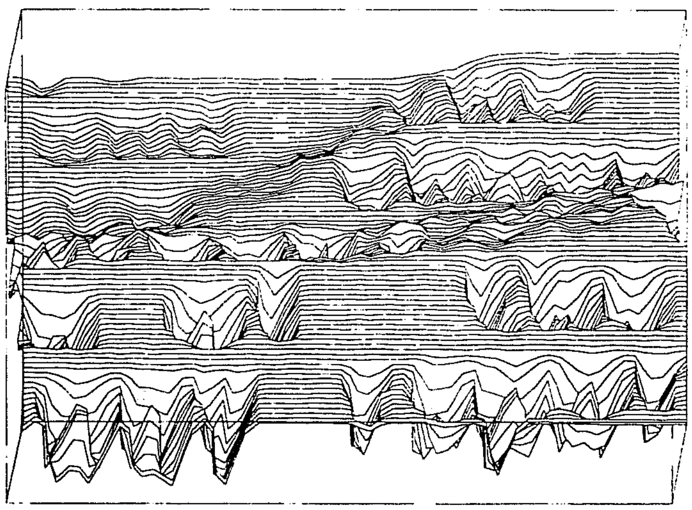


Figure 11—Plot of boxed area in Figure 8. Note the lower contrast of the text in the shadowed region.

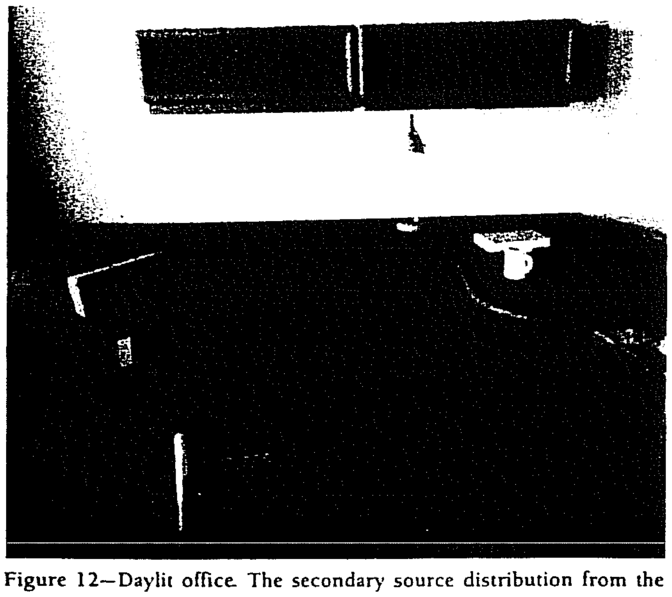


Figure 12—Daylit office. The secondary source distribution from the window and blinds (off left) was precalculated with MAKEDIST, which calls RTRACE.

serious mistakes, and gives the user mobility to find errors visible only from certain vantage points. One may increase the resolution selectively, concentrating on more interesting parts of the image. The same freedom of movement and concentration permits crude lighting and visibility evaluation (eg. contrast and glare). Since a full calculation is being performed at a reduced resolution, absolute accuracy is maintained. Luminance values may be queried at any point in the scene, with instant results. Once the scene is in order and one or more good views have been determined, RPICT may be run overnight to produce high resolution pictures suitable for presentation.

The RADIANCE picture file format uses a floating point representation for greater accuracy. Pictures may be scaled in brightness, resized, anti-aliased, and composited digitally with different filters provided. Drivers display the pictures on monitors or make copies on film, video or paper. Some display programs can superimpose numerical information on images at selected locations for more accurate evaluation.

Figure 10 shows an atrium with luminance values corresponding to a few pixels in the image. The picture actually contains a few hundred-thousand such values, more than could possibly be displayed numerically. This combination of quantitative and qualitative analyses is a critical aid to design. The public domain program IMAGETOOL from the National Center for Supercomputing Applications was used to produce a three-dimensional plot of the picture values from the small boxed area of Figure 8, shown in Figure 11.

Of course, images are not the only output desired from a lighting simulation. RTRACE provides a convenient interface for obtaining other kinds of information from the calculation. Individual radiance values may be computed and combined to get luminance, contrast rendering factors, equivalent sphere illumination, glare indices, or any other lighting metric. Besides radiance, irradiance, intersection point, surface normal direction, and other information are available. Typically, RTRACE is run from a shell script or other program. For example, the utility MAKEDIST calculates a source distribution from a geometric specification by calling RTRACE. This method was used to calculate the light distribution from the window with venetian blinds in Figure 12. By connecting certain display drivers to RTRACE, additional information may be computed interactively at selected image locations, such as the illuminance on a surface or the ray propagation tree.

Validation

RADIANCE has been compared to scale model measurements and different lighting calculation programs in an initial validation study. Figure 13a shows calculated vs measured values for an empty rectangular room with a single Lambertian source. (Since SUPERLITE is primarily a daylighting program, the data were adapted from a skylight model comparison.[9]) Figure 13b compares RADIANCE with Lighting Technologies' PC lighting program, LUMEN-MICRO, for a rectangular room with four fluorescent fixtures.[10] Figure 14a shows calculated and measured values for a rectangular office with a single window under clear sky conditions.[11] Figure 14b shows the same office under cloudy sky condi-

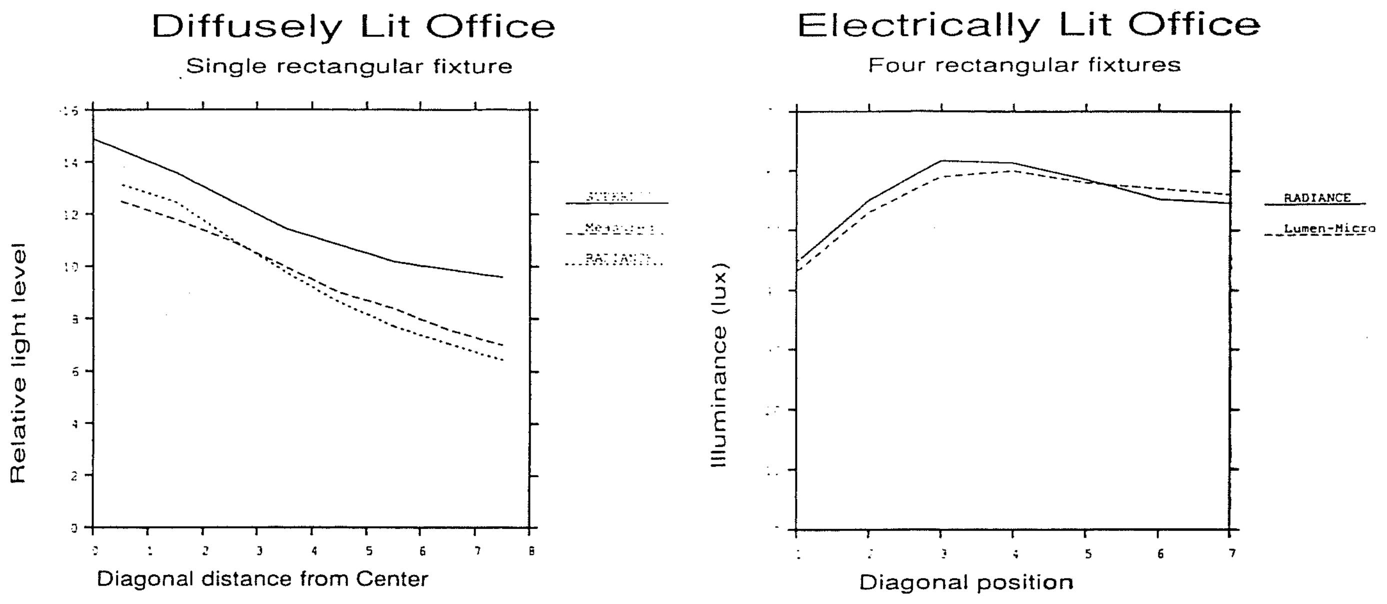


Figure 13—Electric light comparisons

tions. These and other comparisons that have been conducted show good correlation with measured data and conventional lighting simulations. However, RADIANCE is capable of going beyond the limitations of other programs and the simple cases studied here.

The above comparisons were made with empty rooms and diffuse surfaces, exercising only a tiny subset of what RADIANCE can model. The program's accuracy has also been verified for unempty spaces, but more work is needed in the validation of non-Lambertian surfaces. In a semi-specular model, the accurate characterization of surface reflectance properties becomes critical. Current methods require the painstaking measurement of bidirectional distribution functions using a goniophotometer. This approach is impractical given the tremendous variety of architectural materials in use, and a better method must be found for the calculation to be both valid and practical. Fortunately, there is hope of parameterizing reflectance properties and measuring their distribution functions with more conventional hardware, such as video cameras. Until this part of the validation is completed, the reflectance of semi-specular surfaces will only be approximate.

The software has been in use for the past 2 years by the Architecture Department at the University of California at Berkeley, which augments its computer modeling courses with rendering and simulation.

Conclusion

A new software tool for the simulation and visualization of illuminated spaces has been described. The program uses ray tracing to calculate luminance and synthetic images in complicated spaces with diffuse, specular, and transmissive surfaces. The results have compared favorably with other lighting simulations and scale model measurements in an initial

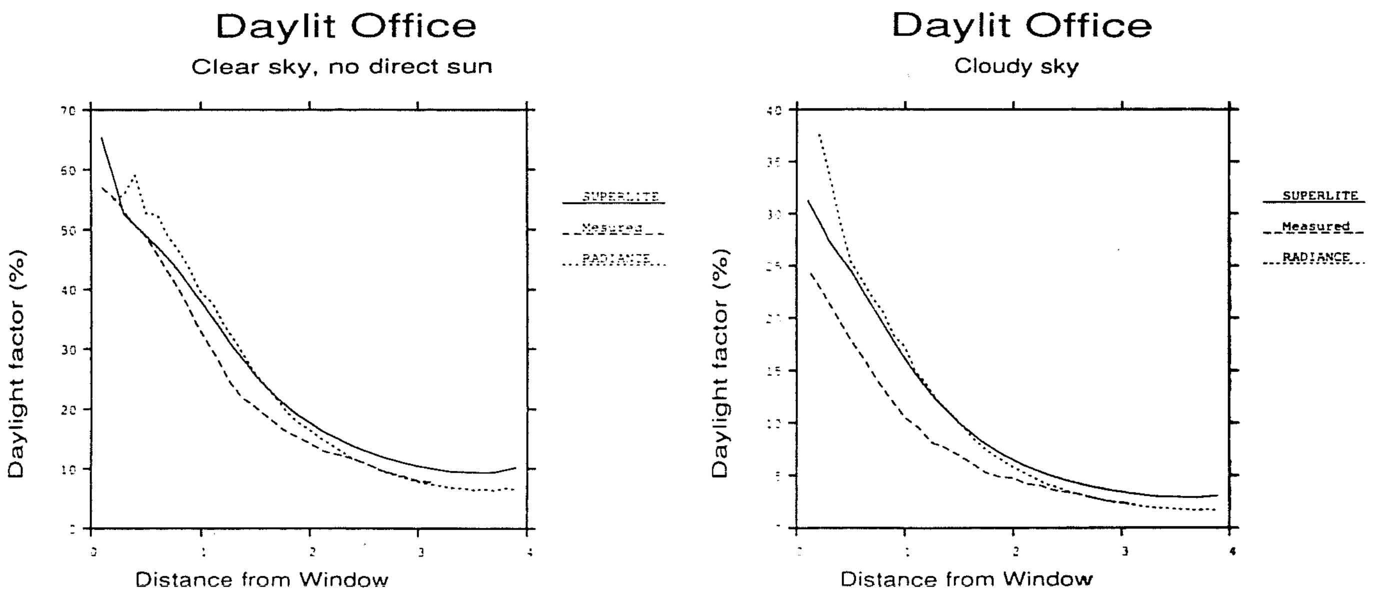


Figure 14—Daylight comparisons

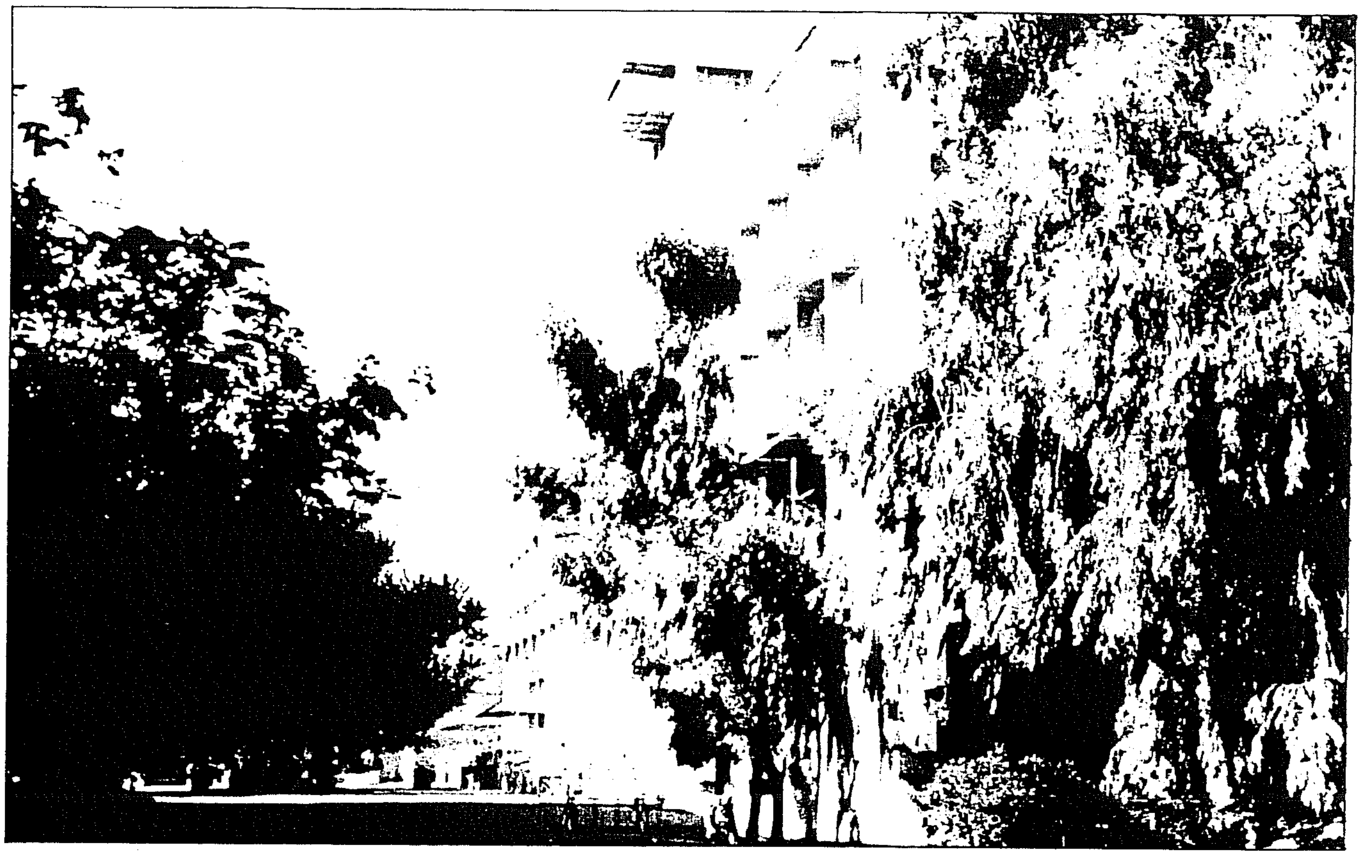
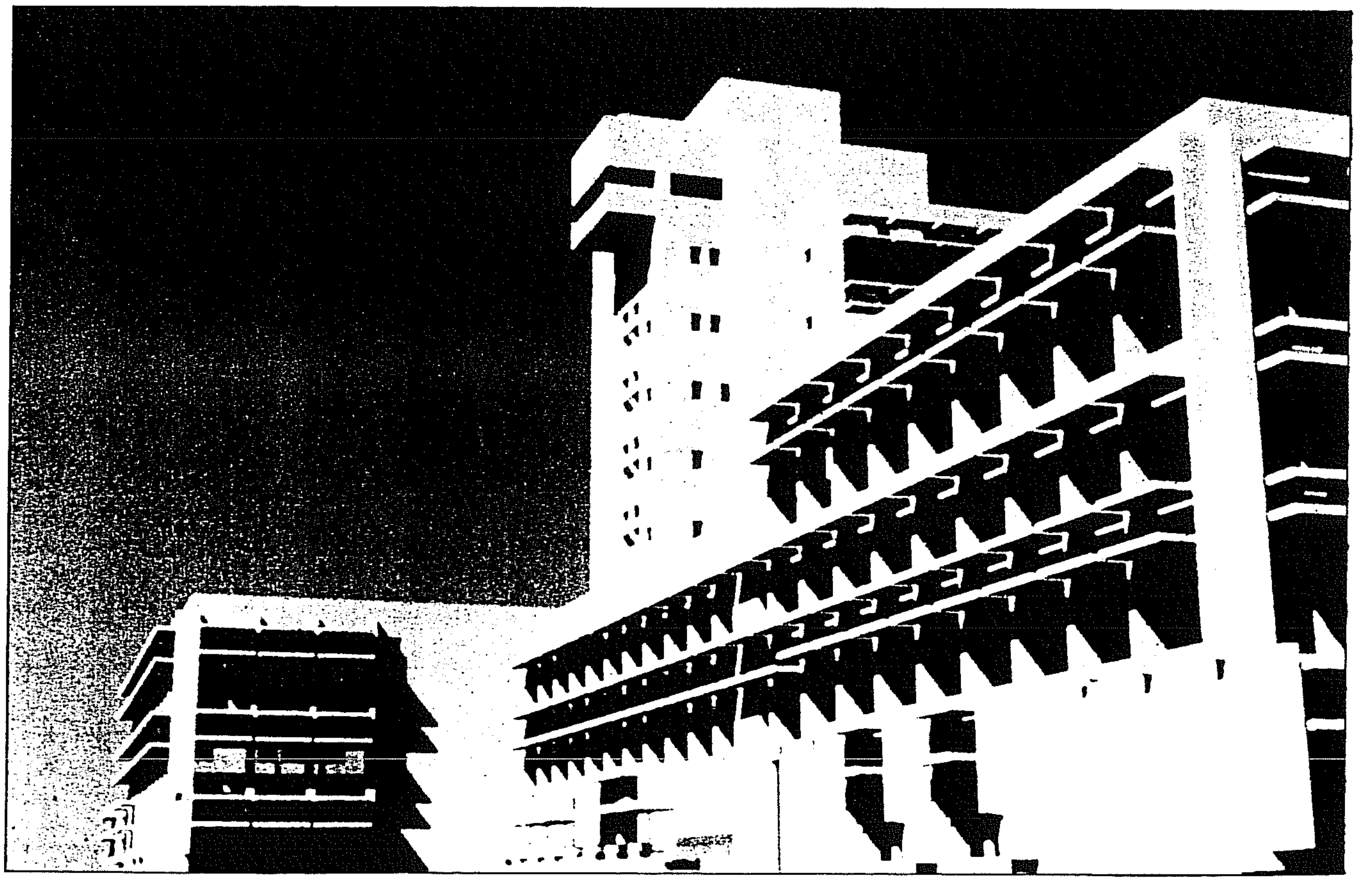


Figure 15—Shows the University of California—Berkeley architecture building, Wurster Hall, (above) and a rendering (below) constructed using McDonnell Douglas's GRAPHICAL DESIGN SYSTEM (GDS). This visualization uses RADIANCE as a back-end renderer to a commercial CAD system, yielding visual output with numerical validity. The model shown contains over 24,000 polygons and took 12 hrs to produce on a Sun 3/60.



validation study. The software is being made available at no charge in hopes that it will stimulate new research in lighting simulation and improve the quality and efficiency of lighting design.

Acknowledgements

Sincere thanks go to Francis Rubinstein for his input and support, and to the students and faculty in the Department of Architecture at UCB.

This work was supported by the Assistant Secretary for Conservation and Renewable Energy, Office of Buildings and Community Systems Energy Research and Development, Buildings Equipment Division of the US Department of Energy under Contract No. DE-AC03-76SF00098. Please write to these guys and tell them to send us some money.

References

1. DiLaura, David L. 1979. On a new technique for interreflected component calculations. *J of the IES* 9(no. 1):53–59

2. DiLaura, David L.; Igoe, Denis P.; and Mistrick, Richard G. 1985. Synthetic photography. *LD+A* 15(no. 8):24–27

3. Whitted, Turner. 1980. An improved illumination model for shaded display. *Communications of the ACM* 23(no. 6):343–349

4. Ward, Gregory J.; and Rubinstein, Francis M. 1988. A new technique for computer simulation of illuminated spaces. *J of the IES* 17(no. 1):80–91

5. Ward, Gregory J.; Rubinstein, Francis M.; and Clear, Robert D. 1988. A ray tracing solution for diffuse interreflection. *Computer Graphics* 22(no. 4):85–92

6. Snyder, John M.; and Barr, Alan H. 1987. Ray tracing complex models containing surface tessellations. *Computer Graphics* 21(no. 4):119–128

7. Frandsen, Sophus. 1987. The scale of light. *International Lighting Review* 87(no. 3):108–112

8. IESNA Computer Committee. 1986. *IESNA recommended standard file format for electronic transfer of photometric data. IESNA*. LM-63-1986.

9. Spitzglas, Mark; Navvab, M.; Kim, J.H.; and Selkowitz, S. 1985. Scale-model measurements for a daylighting photometric database. *J of the IES* 15(no. 1):41–61.

10. Lighting Technologies. 1987. LUMEN-MICRO interior lighting analysis system user's manual version. 4.1: 43–55.

11. Selkowitz, Stephen; Kim, Jong-Jin; Navvab, Mojtaba; and Winkelmann, Frederick. 1982. The DOE-2 and Superlite daylighting programs. Lawrence Berkeley Laboratory Report 14569

Discussion

Backward ray tracing is a viable technique for computer graphics.

1. It is incorrect for the author to say that current software can handle only simple geometry and empty rooms with little more information than an illuminance table (see Miller and Ngai, 1986 IES Conference Proceedings).

2. The technique of handling objects using XFORM is great; the calculation complexity versus number of surfaces is very good news.

3. The handling of transmissive materials is also a plus.

4. The author did not address the psychological aspects of transforming a real scene into a computer graphic image.

In 1984, Miller and Ngai considered backward ray tracing; this technique has many attractive features as stated in the paper such as handling of non-diffuse materials. There are, however, two major drawbacks:

a) Unlike flux transfer calculations, ray tracing is view-specific. Every time a new view of the same scene is needed, an entirely new calculation must be done.

b) The quality of the picture depends on the number of rays traced. In order to get a good picture, a large number of calculations must be done; therefore, computational time is a problem. For example, it took 12 hrs to produce the picture in Figure 15 with a Sun 3/60. Can one use a desktop IBM PC AT, and if so, how long would it take to compute? It is fair to say, however, that as microcomputer performance improves, the time required to generate a picture will decrease.

Peter Ngai
Peerless Lighting

It is refreshing to see the application of a method widely used outside the lighting industry (ray tracing) to solve lighting problems. While ray tracing is used in most image synthesis software, I have never before seen an attempt to test the technique's accuracy as a predictive tool; I am glad to see it done here.

While the author is attempting to serve the lighting community by developing this program and making it available, surely he realizes that the lighting community is not a UNIX-environment running Sun computers. We are a DOS-oriented group running PCs with an occasional MacIntosh (to make slides). Would the author comment on his plans to port RADIANCE to the PC-DOS environment? Also, running on a PC, what might be a typical execution time? Lastly, FORTRAN and C compilers are available for DOS to allow 80286 and 80386 processors to operate in protected, 16MB mode—in what language is RADIANCE written?

Steve Stannard
Consultant to GENLYTE

Response

The author would like to thank the discussors for their comments, which will be addressed in order.

To Mr. Ngai

The author did not mean to imply that currently all lighting software is limited to simple geometries and point calculations of illuminance. However, most commercial programs are restricted either to diffuse surfaces or regular geometries, and only recently has visualization been taken seriously in lighting simulation. Some excellent research has been done by Ngai, Miller, and others. We are hopeful that these techniques and programs will make it into general practice.

As a psychophysiological phenomenon, human vision is difficult to characterize in simple terms. A number of notable efforts have been made to determine the relationship between presentation and perception, but none have been complete-

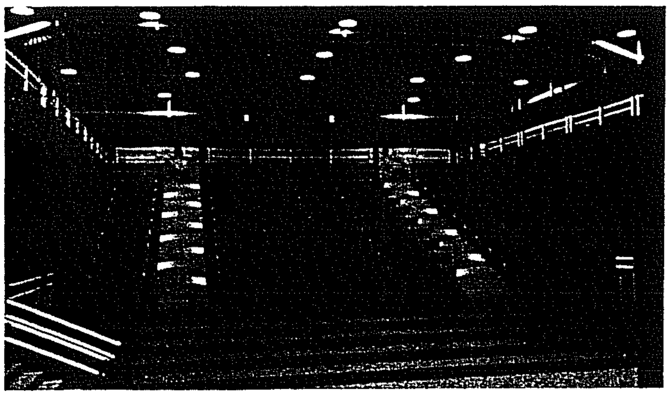
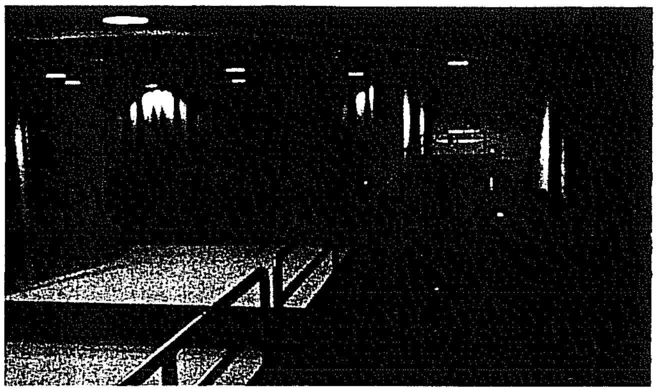



Figure 16—The Candlestick Park Theater. First view shows seating in early model. Second view shows stage after the addition of railings and final material selections. (Courtesy Mark Mack Associates)

ly successful in separating the variables involved. How much of perception and evaluation depends on viewer experience, how much on visual acuity, how much on ambient environment, and so on? Until these factors are isolated, it is not really possible to compensate for variations in any one of them.

Rather than compensating for an inadequate representation of simulation results, we seek first to improve the display medium to the limits available with current technology. If it were possible to reproduce the visible wavefront at full intensity over the entire field of view, no brightness mapping would be required. The user would see and interpret things just as if the environment actually existed. As unlikely as it may seem, this ideal is perhaps not so far away. So-called virtual reality display devices are being developed for interactive computer systems, and the projected cost is comparable to current color monitors.

It is true that parts of the ray tracing calculation are view-dependent. However, the parts that do not depend on view, specifically diffuse reflection, are not recalculated for every image. Just as in flux transfer, illuminance values are reused during ray tracing for many view points, and do not significantly add to the computation time after the first image. It is only because ray tracing models specular reflection, an inherently view-dependent phenomenon, that any recalculation, beyond hidden surface removal, is necessary.

Ngai is correct in his observation that the quality of an image is a function of the number of rays traced, which is directly proportional to the time required on the computer. Intuitively, there is a certain justice here. Should a better picture take more time, less time, or the same amount of time? For evaluation and debugging, turnaround time is more important than image resolution, so it is nice to be able to get output quickly at the expense of visual quality. The calculation may be just as accurate, but fewer luminance values are produced. For presentation and publication, we frequently spend as much computer time as is reasonable to get a nice looking result. On a slower machine, we would probably compromise quality for execution time.

To Mr. Stannard

The decision to develop the software in a UNIX environment was based on practicality. In the beginning, the only machines with sufficient memory and computer power were multi-user systems. By writing under the UNIX operating system, the software could be ported easily between hardware platforms, thus reducing dependence on any one machine architecture.

Although there are personal computers now with significant computing and memory resources, portability is still a problem. In the PC-DOS environment, there must be a half-dozen graphics cards in common use, and many more input devices. Although the hardware of the MacIntosh is somewhat stable, there are still many flavors to support and many technical volumes to read before the first menu appears.

However, the situation is not hopeless for lighting people who cannot or will not spend the additional 30 percent it takes to buy a UNIX workstation nowadays. One hope is that software standardization efforts will drag the PC-DOS community into the world of graphics windows and real operating systems. UNIX is already an option for the Mac II and some 80386 machines. Another hope is that some enterprising individual will take the C source code for RADIANCE and cram it onto an unsuspecting personal computer. Finally, there is a small weak hope that the author will acquire the funding necessary to do it himself.

There is no answer to the question of typical execution time. The interactive renderer, RVIEW, is progressive in its calculation, and is the program most frequently used by designers. The batch rendering program, RPICT, can take anywhere from a minute to several hours to compute an image, depending on the resolution requested and the scene complexity. Typically, a user will do whatever is possible in a given amount of time.

Availability

To get a copy of RADIANCE, send a 1/4-inch tape cartridge (300 ft or longer) with a prepaid return envelope to:

Greg Ward
Building 90, Room 3111
Lawrence Berkeley Laboratory
1 Cyclotron Road
Berkeley, CA 94720

The author: Greg Ward is a staff scientist at Lawrence Berkeley Labortory He graduated in physics from the University of California—Berkeley and earned his master's in computer science from San Francisco State University. This paper was originally presented at the 1989 IES Annual Conference.